GRWORKBENCH: A COMPUTATIONAL SYSTEM BASED ON DIFFERENTIAL GEOMETRY

SUSAN M. SCOTT, BENJAMIN J. K. EVANS AND ANTONY C. SEARLE
Department of Physics and Theoretical Physics, Faculty of Science,
The Australian National University, Canberra ACT 0200 Australia
E-mail: Susan.Scott@anu.edu.au, Ben.Evans@anusf.edu.au, Antony.Searle@anu.edu.au

We have developed a new tool for numerical work in General Relativity: GRworkbench. While past tools have been *ad hoc*, GRworkbench closely follows the framework of Differential Geometry to provide a robust and general way of computing on analytically defined space-times. We discuss the relationship between Differential Geometry and C++ classes in GRworkbench, and demonstrate their utility.

1 Introduction

We have developed a new class of computational tool for General Relativity. Previous tools have fallen into three categories; large scale simulations that evolve space-times from initial conditions, symbolic manipulators, and *ad hoc* numerical systems.

GRworkbench[1] uses numerical variants of standard differential geometric entities to rigorously define space-times in a fashion amenable to computation. This system forms a strong base on which to build generally applicable numerical algorithms, capable of acting on any space-time for which a basic analytic definition is available.

This paper will focus on GRworkbench's roots in differential geometry and will demonstrate the software's wide applicability to problems via consideration of a specific example—geodesic tracing in the Schwarzschild space-time. For a discussion of numerical algorithms, visualization techniques and the user interface employed by GRworkbench, see Evans[2] and Searle[3].

2 Discrete differential geometric structure

We follow the conventions of Hawking and Ellis[4]: A C^r *n-dimensional manifold* $\mathcal{M}$ is a set $\mathcal{M}$ together with a C^r *atlas* $\{(\mathcal{U}_\alpha, \phi_\alpha)\}$, that is to say a collection of charts $(\mathcal{U}_\alpha, \phi_\alpha)$ where the $\mathcal{U}_\alpha$ are subsets of $\mathcal{M}$ and the ϕ_α are one-to-one maps of the corresponding $\mathcal{U}_\alpha$ to open sets in $\mathbb{R}^n$ such that

1. the $\mathcal{U}_\alpha$ cover $\mathcal{M}$, i.e. $\mathcal{M} = \bigcup_\alpha \mathcal{U}_\alpha$,
2. if $\mathcal{U}_\alpha \cap \mathcal{U}_\beta$ is non-empty, then the map
$$\phi_\alpha \circ \phi_\beta^{-1} : \phi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \to \phi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)$$
is a C^r map of an open subset of $\mathbb{R}^n$ to an open subset of $\mathbb{R}^n$.

It is assumed, as in Hawking and Ellis, that we are dealing with *paracompact, connected, C^∞ Hausdorff manifolds without boundary.*

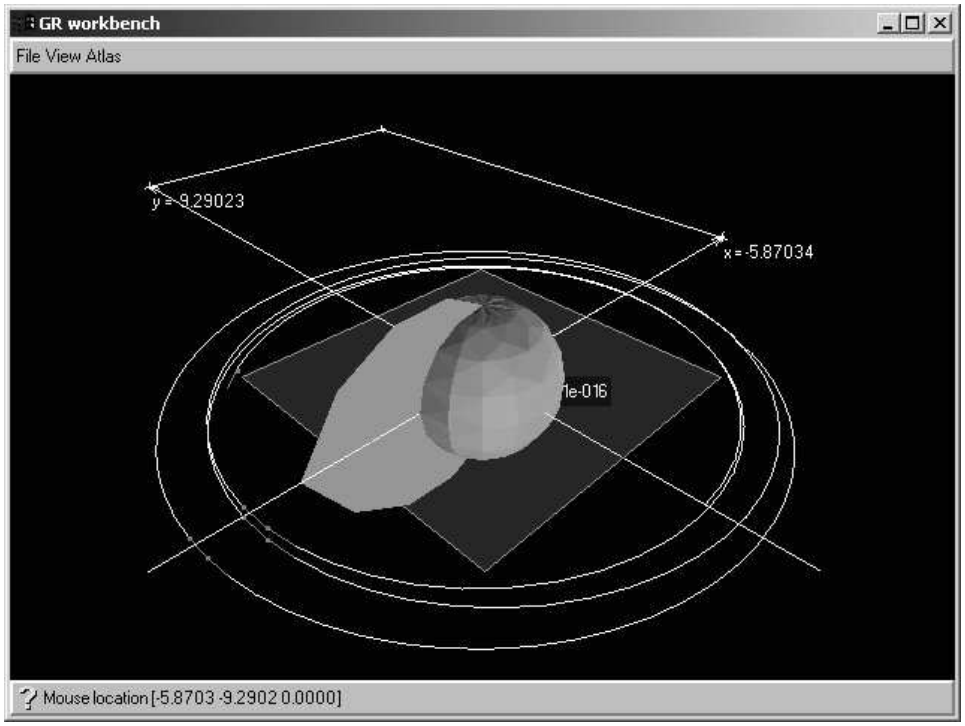

Figure 1. A precessing orbit near the event horizon of a Schwarzschild black hole ($M = 0.5$), computed and visualized in GRworkbench. The flat protrusion from the event horizon is the $\phi = 0, 2\pi$ boundary of the spherical polar chart.

A computer cannot numerically represent the set $\mathcal{M}$—that is, it cannot represent $\mathcal{M}$ using numbers. Nor can it so represent $\mathcal{U}_\alpha$ or the mapping ϕ_α for any chart $(\mathcal{U}_\alpha, \phi_\alpha)$. The manifold, subsets of the manifold, and functions on the manifold are *abstract* entities; the computer cannot deal with them *numerically*. This is not to say that computers cannot deal with these abstract entities *symbolically*—there exist numerous symbolic manipulators, such as Mathematica[5] and Sheep[6], for this purpose.

Computers can, however, operate numerically in $\mathbb{R}^n$. The set $\phi_\alpha(\mathcal{U}_\alpha)$ can be represented numerically, as can the function $\phi_\alpha \circ \phi_\beta^{-1} : \phi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \to \phi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)$. Computers represent real numbers as *floating-point* numbers. This is analogous to base-2 scientific notation, where (a, b) represents $a \times 2^b$. On a modern computer, real numbers are typically represented using 64 bits, meaning that the computer can represent up to $2^{64} \approx 10^{19}$ rational numbers, spaced approximately logarithmically along the real line. This means that, strictly speaking, all sets representable by the computer are closed, compact and totally disconnected. Continuity cannot be sensibly defined for functions on a totally disconnected domain, and many functions, such as $y = \frac{2}{3}x$, are, surprisingly, *not* one-to-one. In this example, two numbers adjacent in the representation, when multiplied by $\frac{2}{3}$, may both produce the same

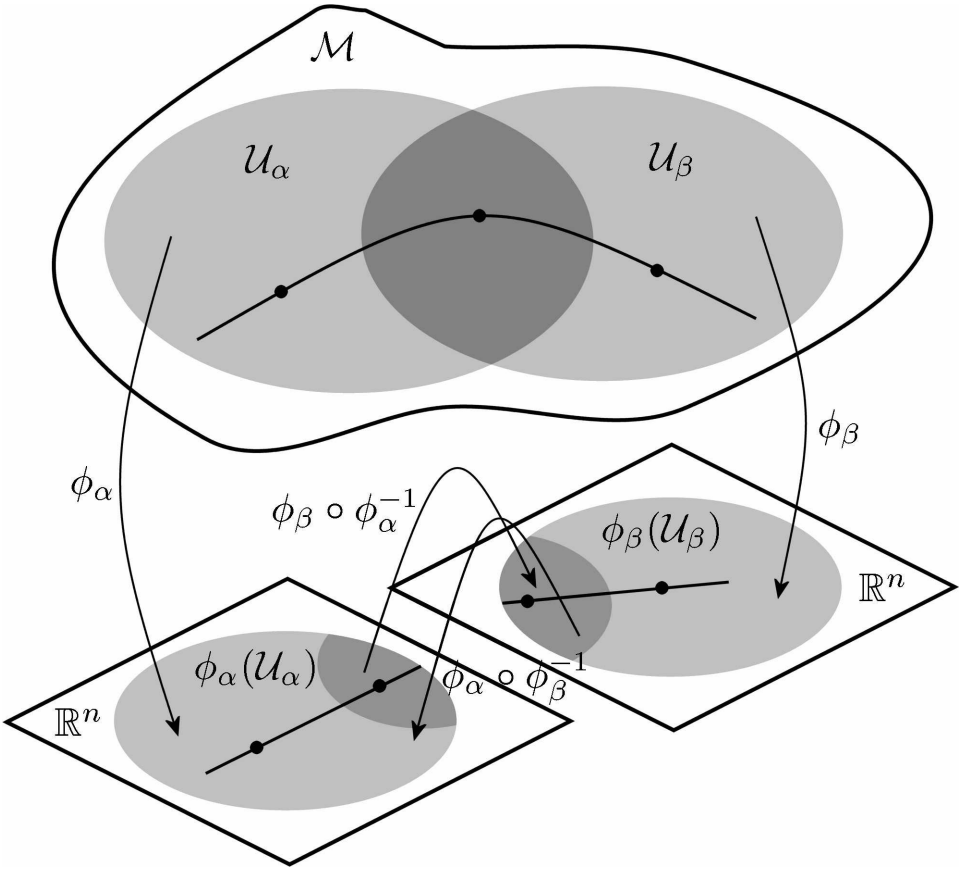

Figure 2. Schematic depiction of a geodesic crossing from the chart $(\mathcal{U}_\alpha, \phi_\alpha)$ to the chart $(\mathcal{U}_\beta, \phi_\beta)$.

floating-point result, as the difference between the true results is less than the precision of the discrete representation.

Thus, although a computer cannot directly represent a manifold, atlas or chart as defined above, it can produce *similar* objects, which we distinguish from the abstract entities by use of a Sans Serif typeface.

We define an Atlas, the numerical representation of an atlas $\{(\mathcal{U}_\alpha, \phi_\alpha)\}$, as a collection of Charts, the numerical representations of charts $(\mathcal{U}_\alpha, \phi_\alpha)$. We define the Chart representing $(\mathcal{U}_\alpha, \phi_\alpha)$ as:

1. The numerical representation of the set $\phi_\alpha(\mathcal{U}_\alpha)$.
2. The set of numerical representations of functions mapping from the Chart to other Charts: $\{\phi_\beta \circ \phi_\alpha^{-1} : \phi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \to \phi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)\}$.

We cannot define a Manifold as an Atlas combined with the set of equivalence classes of points $x \in \phi_\alpha(\mathcal{U}_\alpha)$ under the mappings $\phi_\beta \circ \phi_\alpha^{-1}$, as these mappings are not, in general, one-to-one, due to the discrete representation employed by the computer.

It is possible that $\phi_\alpha \circ \phi_\beta^{-1}(\phi_\beta \circ \phi_\alpha^{-1}(x)) \neq x$, though the difference should be "small", that is, on the order of the local resolution ϵ of the discrete representation. In general, there appears to be no sensible definition for a Manifold, and we do not adopt one.

Although we cannot construct an authoritative set $\mathcal{M}$ using the above procedure, we can still produce a *useful* definition of a Point as the numerical representation of $p \in \mathcal{M}$. First define a Coordinate[a] by:

1. The numerical representation (Chart) of a chart α (abbreviated notation for $(\mathcal{U}_\alpha, \phi_\alpha)$).
2. The numerical representation of a point $x \in \phi_\alpha(\mathcal{U}_\alpha)$.

Note that the numerical representation of x is simply an n-tuple of real numbers, so it is necessary to additionally specify the chart in order to give those numbers the context of a mapping. We can identify a Coordinate (α, x) with a point of the manifold, $\phi_\alpha^{-1}(x) \in \mathcal{U}_\alpha \subseteq \mathcal{M}$. We do not, however, have a numerical representation of the function ϕ_α^{-1}, so to define a Point representing $p \in \mathcal{U}_\alpha$ we use a set of Coordinates:

1. The Coordinate $(\alpha, \phi_\alpha(p) = x)$.
2. The Coordinates $\{(\beta, \phi_\beta \circ \phi_\alpha^{-1}(\phi_\alpha(p))) : \beta \neq \alpha\}$.

Note that these are *not* the Coordinates $(\beta, \phi_\beta(p))$, though the difference should be "small".

Thus, although we can define a Point, its definition is tied to its coordinates under a particular chart. If we were to produce Coordinates $(\alpha, \phi_\alpha(p))$ and $(\beta, \phi_\beta(p))$ for the same point $p \in \mathcal{U}_\alpha \cap \mathcal{U}_\beta \subseteq \mathcal{M}$, the Points manufactured from them would not, in general, consist of precisely the same Coordinates (condition 2 above), and thus would *not* be equal so far as the computer is concerned. We cannot produce a chart-independent representation of a point; we can only produce a chart-dependent Point that is "approximately" chart-independent, in the sense that the difference between the representations arising from different charts is "small".

There is an obvious similarity between the Atlas and Point; both are collections of *different* numerical representations of the *same* abstract object. This abstract layer identifying different numerical representations imparts (approximate) *chart-independence* to numerical operations performed within the structure. Essentially, in the midst of performing a computation the computer can change charts as required by mapping the pertinent parameters to another Chart.

An example we will follow throughout this paper is the computation of a discrete approximation to a geodesic. A geodesic $\gamma : \mathbb{R} \to \mathcal{M}$ may, for some interval around $\tau \in \mathbb{R}$, lie in the domain $\mathcal{U}_\alpha$ of a chart α. It can be represented numerically by the series of Coordinates (and thus by the series of Points generated from the Coordinates) $(\alpha, \phi_\alpha(\gamma(\tau))), (\alpha, \phi_\alpha(\gamma(\tau + \delta))), (\alpha, \phi_\alpha(\gamma(\tau + 2\delta))), \ldots$ for some *step-size* $\delta > 0$, but it is possible that for some $k \in \mathbb{N}$, $\gamma(\tau + (k+1)\delta) \notin \mathcal{U}_\alpha$. In this case, for δ chosen to be sufficiently small, there exists some Chart β such that

[a] Internally named a Node by GRworkbench for historical reasons.

$\gamma(\tau+(k+1)\delta) \in \mathcal{U}_\beta$, and $\gamma(\tau+k\delta) \in \mathcal{U}_\alpha \cap \mathcal{U}_\beta$. Without the abstract identification of the Coordinate $(\alpha, \phi_\alpha(\gamma(\tau+k\delta)))$ with the Coordinate $(\beta, \phi_\beta \circ \phi_\alpha^{-1}(\phi_\alpha(\gamma(\tau+k\delta))))$ via the Point representing the point $\gamma(\tau+k\delta)$, there would be no way in which the next Point, representing the point $\gamma(\tau+(k+1)\delta)$, could be computed. With the identification, however, we can continue to compute the numerical representation from algorithms operating on Chart β. This procedure is depicted schematically in Figure 2.

The interest of users in entities, such as geodesics, that are structured sets of Points, motivates us to define the Object. An Object is a set of Points. The Points constituting an Object will have Coordinates. We define a Segment for a certain Object and Chart as the *image* of the set represented by the Object on that Chart. As such, a Segment is a set of Coordinates.

To use the differential geometric framework to represent a space-time, we must provide one additional component. We extend the definition of a Chart to encompass the provision of a *metric* $\mathbf{g}$—a $(0,2)$ symmetric tensor field on $\mathcal{M}$. We require that each Chart provide a function returning the tensor components $g_{ij}|_x$ with respect to the chart's coordinate basis $\{\partial/\partial x^a\}$, for any coordinate $x \in \phi_\alpha(\mathcal{U}_\alpha)$.

3 Implementation

The numerical differential geometric system defined above may be naturally expressed as a collection of *classes* in the C++[7] programming language. A class is itself a collection of data and related functions. The notation A::B indicates that B is a *member* of class A.

The Atlas class stores chart-independent data and a list of the Charts $\{(\mathcal{U}_\alpha, \phi_\alpha)\}$ that comprise it. In the case of the Atlas representing the Schwarzschild space-time, the Atlas stores the Schwarzschild mass M that defines certain properties of the Charts, such as mandating that the radius $x_1 > 2M$ for exterior charts.

The Chart representing $(\mathcal{U}_\alpha, \phi_\alpha)$ must provide

1. The set $\phi_\alpha(\mathcal{U}_\alpha) \subseteq \mathbb{R}^n$.
2. The functions $\{\phi_\beta \circ \phi_\alpha^{-1} : \phi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta) \to \phi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta)\}$.
3. The metric tensor $g_{ij}|_x$, for $x \in \phi_\alpha(\mathcal{U}_\alpha)$.

For maximum flexibility, we require the user to define the set $\phi_\alpha(\mathcal{U}_\alpha)$ in terms of a *Boolean* function on $\mathbb{R}^n$, that is, a function Chart::Interior : $\mathbb{R}^n \to \{\text{true}, \text{false}\}$.

$$\text{Chart::Interior}(x) = \begin{cases} \text{true if } x \in \phi_\alpha(\mathcal{U}_\alpha) \\ \text{false otherwise} \end{cases} \tag{1}$$

This formalism allows the user to plug in any algorithm to define the domain of the Chart.

For an exterior spherical polar chart of the Schwarzschild space-time, the Chart::Interior function is defined as

$$\text{Chart::Interior}(x) = \begin{cases} \text{true if } x_1 > 2M \text{ and } 0 < x_2 < \pi \text{ and } 0 < x_3 < 2\pi \\ \text{false otherwise} \end{cases} \tag{2}$$

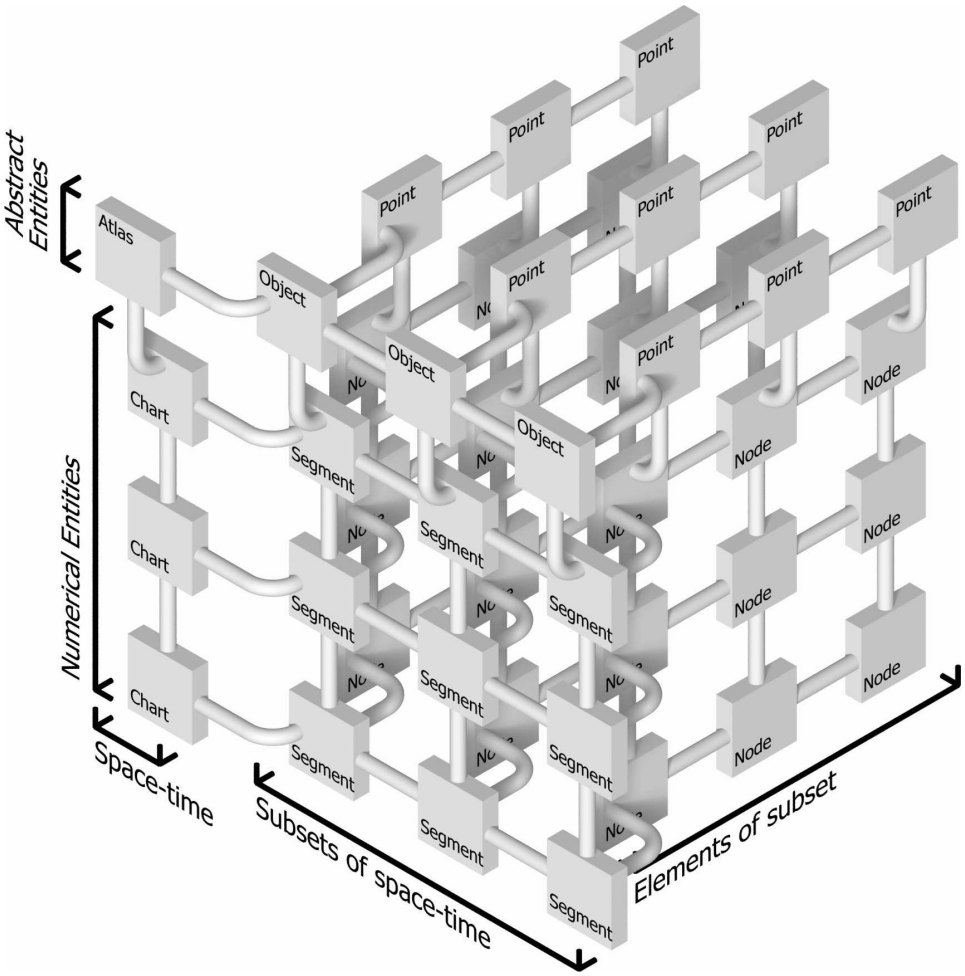

Figure 3. Differential geometric structure of a manifold as encoded by GRworkbench. The entities labelled Nodes are generalisations of the Coordinate concept that may additionally carry a vector and/or scalar.

Maps to other charts, $\phi_\beta \circ \phi_\alpha^{-1}$, are implemented as $\mathbb{R}^n \to \mathbb{R}^n$ functions. An obvious disadvantage of the need to define maps $\phi_\beta \circ \phi_\alpha^{-1}$ rather than simply maps ϕ_α is that, in general, an Atlas comprising m Charts requires up to $m(m-1)$ $\mathbb{R}^n \to \mathbb{R}^n$ maps rather than just the m $\mathcal{M} \to \mathbb{R}^n$ maps. This number can be reduced by permitting the implicit definition of maps, by application of several maps between other charts, such as the map $\phi_\beta \circ \phi_\alpha^{-1} = (\phi_\beta \circ \phi_\gamma^{-1}) \circ (\phi_\gamma \circ \phi_\alpha^{-1})$: $\phi_\alpha(\mathcal{U}_\alpha \cap \mathcal{U}_\beta \cap \mathcal{U}_\gamma) \to \phi_\beta(\mathcal{U}_\alpha \cap \mathcal{U}_\beta \cap \mathcal{U}_\gamma)$.

Two spherical polar charts are required to cover the entire exterior ($x_1 > 2M$) region of the Schwarzschild space-time. Choosing the second spherical polar chart

so that its polar axis corresponds to the coordinates[b] $x_2 = \pi/2$ and $x_3 = \pi/2, 3\pi/2$ allows us to define the map between the charts as:

$$\begin{aligned} x'_0 &= x_0 \\ x'_1 &= x_1 \\ x'_2 &= \arg(\sin x_2 \sin x_3 + \sqrt{\sin^2 x_2 \sin^2 x_3 - 1}) \\ x'_3 &= \arg(-\sin x_2 \cos x_3 + \sqrt{-1} \cos x_2) \end{aligned} \tag{3}$$

This map is its own inverse.

The numerical representation of a differential geometric structure cannot know, in advance, if that representation is complete. That is, the implementation cannot determine if the Atlas provided covers the entire manifold $\mathcal{M}$, as the implementation has no *a priori* knowledge of the manifold $\mathcal{M}$ itself. Neither can the implementation determine if the $\mathbb{R}^n \to \mathbb{R}^n$ maps supplied by the Charts are comprehensive, as the relationships between Charts are defined solely in terms of these maps. This allows the user freedom to implement systems for which the Atlas provided does *not* cover the manifold, and for which the $\mathbb{R}^n \to \mathbb{R}^n$ maps supplied by the Charts are *not* comprehensive.

This is not cause for concern. When the implementation cannot successfully continue correct computation due to the lack of a Chart or map, the algorithm halts. If the user wishes to perform the computation, they must add a new component to the incomplete system. In many circumstances, though, the user may be concerned only with a portion of the system. For example, to study orbits in a Schwarzschild space-time, it is sufficient to define only *exterior* charts. To require the user to define portions of the space-time which they have no intention of utilizing—in this case, the interior Schwarzschild space-time—would be to unnecessarily inconvenience the user.

To complete the definition of the Chart, the user must supply a function returning the metric tensor components $g_{ij}|_x$, for $x \in \phi_\alpha(\mathcal{U}_\alpha)$. For any exterior spherical polar chart of the Schwarzschild space-time, the components are defined as:

$$\begin{array}{llll} g_{00}|_x = -1 + \frac{2M}{x_1} & g_{01}|_x = 0 & g_{02}|_x = 0 & g_{03}|_x = 0 \\ g_{10}|_x = 0 & g_{11}|_x = (1 - \frac{2M}{x_1})^{-1} & g_{12}|_x = 0 & g_{13}|_x = 0 \\ g_{20}|_x = 0 & g_{21}|_x = 0 & g_{22}|_x = (x_1)^2 & g_{23}|_x = 0 \\ g_{30}|_x = 0 & g_{31}|_x = 0 & g_{32}|_x = 0 & g_{33}|_x = (x_1)^2 \sin^2 x_2 \end{array} \tag{4}$$

In the current implementation, Atlases and Charts are defined by the user by modifying trivial C++ code in supplied "template" files[c]. Only the most basic programming skills are required of the user. Compared to an obvious alternative, namely, using a scripting language, this method has the advantage of producing efficient, pre-compiled code, but the disadvantage of *relinking* the *executable* when space-time definitions are modified. Such modifications are, however, comparatively rare for common usage.

The remaining classes, Objects, Segments, Points and Coordinates are implemented in a straightforward way. Coordinates contain an n-tuple of floating point values and an indirect reference (via their containing Segment) to their Charts. Any

[b] We use the C++ convention that indices begin from 0.
[c] Not to be confused with the C++ template keyword.

Point owns a list of its Coordinate representations on all applicable Charts. A Segment similarly maintains a list of the Coordinates of the Points of a particular Object on a particular Chart. An Object maintains a list of its Points, and Segments on various Charts, and a Chart maintains a list of Segments on it from various Objects. Segments and Coordinates thus belong to two lists, which are represented perpendicularly to one another in Figure 3.

The above components are genuinely sufficient to define a space-time. Where GRworkbench requires certain information to perform a task, such as the geodesic equation to compute a geodesic, it extracts that information numerically using the relevant definition. For example, the geodesic equation is given by

$$\frac{\mathrm{d}^2 x^a}{\mathrm{d}\tau^2} = -\Gamma^a{}_{bc} \frac{\mathrm{d}x^b}{\mathrm{d}\tau} \frac{\mathrm{d}x^c}{\mathrm{d}\tau}, \tag{5}$$

where the Christoffel symbol Γ is given by

$$\Gamma^a{}_{bc} = \frac{1}{2} g^{ad} \left(\frac{\partial}{\partial x^c} g_{db} + \frac{\partial}{\partial x^b} g_{dc} - \frac{\partial}{\partial x^d} g_{bc} \right). \tag{6}$$

We may compute g^{ab} as the inverse matrix of g_{ab}, and compute the partial derivatives $\frac{\partial}{\partial x^c} g_{ab}$ by numerical differentiation of g_{ab}. Thus, GRworkbench can compute the geodesic equation directly from the user-supplied metric for any given space-time.

4 Utility

Using the definition for a Schwarzschild Atlas given in Section 3, consisting of two exterior spherical polar Charts, we can begin to trace geodesics, such as those of Figure 1, on the space-time *without deriving the geodesic equations*.

A well-known fact about the Schwarzschild space-time is that there exists a circular null orbit at $x_1 = 3M$. Although this fact is not immediately apparent analytically, it can be reproduced using GRworkbench in an *experimental* mode.

We release geodesics from $x_0 = 0, x_2 = \pi/2, x_3 = \pi/2$ for varying values of $x_1 > 2M$, and mandate that $\frac{\partial x^0}{\partial \tau} > 0, \frac{\partial x^1}{\partial \tau} = \frac{\partial x^2}{\partial \tau} = 0, \frac{\partial x^3}{\partial \tau} > 0$ and the geodesic be null. GRworkbench then uses the metric to produce an initial null tangent vector satisfying these conditions[d]. For any x_1, the geodesic will either escape to infinity, or fall into the event horizon. Using these conditions, we can iterate down on the value of x_1 that divides these two types of behaviour. Figure 4 shows the geodesics traced in this process. As expected, the value below which geodesics fall into the event horizon, and above which geodesics escape to infinity, is given by $x_1 = 1.5 = 3 \times 0.5 = 3M$ to high accuracy. The computations were performed interactively in a few minutes on an inexpensive notebook computer.

While this is a trivial example, recall that this functionality is available purely from a minimal definition of the space-time—and as such, is available for *any* space-time that can be so defined. Additionally, there are many algorithms implemented,

[d] GRworkbench does this by breaking the given vector into a purely time-like and purely space-like component, and re-scaling these parts so that they sum to a null vector.

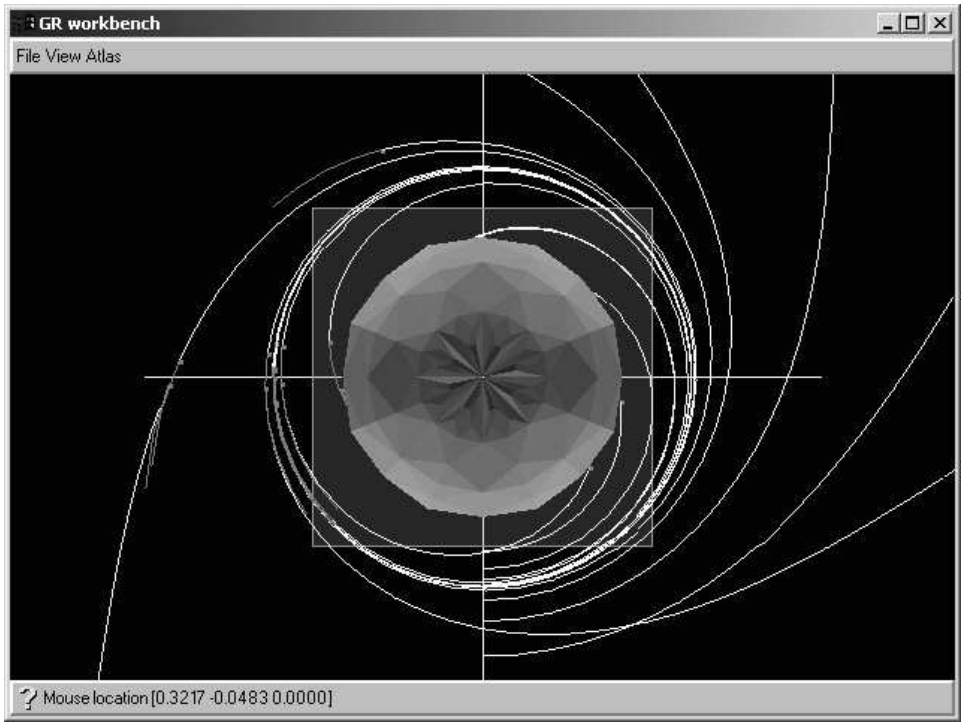

Figure 4. Null geodesics in the Schwarzschild space-time, iterating in on the $r = 1.5 = 3M$ circular null orbit.

beyond the featured example of geodesic tracing, that operate with similar flexibility; from the production of null cones to the examination of causal connections, to the computation of proper distances between points on a space-like hypersurface. GRworkbench's firm basis in differential geometry makes it a truly general tool, and encourages a new *experimental* approach to problems in General Relativity—one of trial and observation.

5 Conclusion

GRworkbench successfully implements a numerical analogue of a standard differential geometric system, mirroring a manifold and atlas of charts with the C++ classes Atlas and Chart. Provision of a metric on a Chart completes the definition of a space-time in General Relativity.

From this minimal definition, complex algorithms, such as geodesic tracing, may be "bootstrapped" through layers of numerical operations, to produce useful results, in novel ways, with minimal effort. As such, GRworkbench permits, and encourages, an "experimental" approach to problem-solving in General Relativity.

References

1. http://grworkbench.anu.edu.au/
2. Evans B J K 2000 *New Geometric Analysis Tools for Investigating Global Structure in General Relativity* (Doctoral Thesis) The Australian National University.
3. Searle A C 1999 *GRworkbench* (Honours Thesis) The Australian National University.
4. Hawking S W and Ellis G F R 1973 *The Large Scale Structure of Space-time* (Cambridge University Press)
5. http://www.wolfram.com/products/mathematica
6. ftp://ftp.maths.qmw.ac.uk/pub/sheep/
7. Stroustrup B 1997 *The C++ Programming Language* 3rd Ed. (Addison-Wesley)